Inconsistences in Hybrid Knowledge Bases

Daria Stepanova
Institute of Information Systems, Vienna University of Technology
Favoritenstraße 9-11, A-1040 Vienna, Austria
dasha@kr.tuwien.ac.at

Abstract

Hybrid knowledge bases (KBs) are a well-known paradigm for combining nonmonotonic rules and description logic ontologies, which gained increasing attention in the past years. Such hybrid KBs have the potential of becoming very powerful tools for knowledge representation and reasoning. However, the interaction between rules and ontologies might cause undesired inconsistencies, making the whole system unusable. The research area in which the thesis will be stated is devoted to the investigation of the notion of inconsistencies and development of the methodology for dealing with them within hybrid KBs.

1 Introduction and Problem Description

The increasing popularity of the World Wide Web and distributed systems has created the need for hybrid formalisms that combine nonmonotonic rules and description logic ontologies. Various approaches for such combination have been proposed (see de Bruijn et al. 2009) for overview).

In this thesis we focus on Description Logic (DL-) programs (Eiter et al. 2008), which couple rules and ontologies in a loose manner. The data sources in DL-programs are treated separately, but a bidirectional information flow between them is arranged. This makes them highly expressive systems capable of solving complex reasoning problems on top of ontologies. The interaction between rules and ontologies, however, might give rise to undesired inconsistencies, making the whole system broken and thus unusable.

Unfortunately, currently available systems suffer from the inability to deal with inconsistencies with ease, which forms a major obstacle for their wider acceptance. The development of a sophisticated framework for handling inconsistencies in DL-programs is therefore the overall goal of this PhD project.

2 State of the Art

Description Logics (DLs) are applied as a mature logical formalism, which is focused on specifying and reasoning in relation to conceptual knowledge. Rules in the sense of logic programming are a well-accepted paradigm for declarative problem solving designed to target issues associated with nonmonotonic inference.

Many applications, however, require the combination of features of both DLs and rules. Thus, the natural solution of combining these formalisms has given rise to the notion of hybrid KBs and study DL-programs (Eiter et al. 2008) as a prominent example which realizes this approach.

Example 1 Consider the DL-program $\Pi$ in Figure 1, representing information about children of a primary school and their parents. It consists of an ontology $O$ with a taxonomy $T$ of concepts in (1)-(3) and a data part (i.e., assertions) $A$ about some individuals in (4)-(6). The rules $P$ contain facts (7), (8) and logic rules: (9) identifies fathers from the ontology, upon some feeding information; (10) checks the constraint that a child has for sure at most one father, unless it is adopted.

The bidirectional flow of information in the DL-programs between the logic program and the DL ontology is achieved via so-called DL-atoms. For instance, a DL-atom

\[ O = \{(1) \text{Child} \sqsubseteq \exists \text{hasParent}, (2) \text{Adopted} \sqsubseteq \text{Child}, (3) \text{Female} \sqsubseteq \lnot \text{Male}, (4) \text{Male} \text{(pat)}, (5) \text{Male} \text{(john)}, (6) \text{hasParent} \text{(john, pat)}\} \]

\[ P = \{(7) \text{ischildof} \text{(john, alex); (8) boy(john); (9) hasfather} \text{(X, Y)} \leftarrow \text{DL[Male} \sqcup \text{boy; Male} \text{]}(Y) \text{DL[hasParent]}(X, Y)\}; (10) \bot \leftarrow \text{not DL[Adopted]}(X), Y_1 \neq Y_2, \text{hasfather} \text{(X, Y_1), ischildof} \text{(X, Y_2), not DL[Child} \sqcup \text{boy; } \lnot \text{Male}(Y_2); \} \]

Figure 1: DL-program $\Pi$ over a family ontology
A recent work (Denecker, Bruynooghe, and Vennekens 2012) which generates minimal inconsistent subsets and removes work we look at weak and flp-answer sets (see (Eiter et al. which are minimal models of the program reduct. In this vulnerable for contradiction than individual representations. In general, combining different pieces of knowledge is more inconsistencies in Combination of Logical Formalisms

The efforts towards detecting and solving inconsistencies in logic rules are mostly described in the papers that focus on debugging of logic programs (Gebser et al. 2008; Syrjänen 2006). The approach by Syrjänen (Syrjänen 2006) addresses the issue of debugging incoherent logic programs, which is adapted from the field of symbolic diagnosis (Reiter 1987). A generalization of the same problem is given in the work (Gebser et al. 2008), which provides explanations why interpretations are not answer sets of a program under consideration. The consistency-restoring rules of Balduccini and Gelfond (Balduccini and Gelfond 2003) form another related approach in this regard.

In the area of paraconsistent reasoning for logic programs the most prominent works include (Przymusinski 1991; Sakama and Inoue 1995). For more discussion see (Eiter, Leone, and Saccà 1997). In the work (Eiter, Fink, and Moura 2010) a semantic characterization of semi-stable models in terms of bi-models and of semi-equilibrium models is given. A recent work (Denecker, Bruynooghe, and Vennekens 2012) surveys and explains the application of the approximation fixpoint theory to the semantics of logic programming and answer set programming and generalizations of these.

Inconsistencies in Combination of Logical Formalisms

In general, combining different pieces of knowledge is more vulnerable for contradiction than individual representations.
4 Preliminary Results Accomplished

In our research we assume that the ontology and the rules of the DL-program are consistent when considered separately and the inconsistencies arise as a result of their combination. The reasons for inconsistencies therefore lie in the wrong values of some DL-atoms occurring in the DL-program.

We started the work by identifying DL-atoms which have always the same value regardless of the ontology and interpretation of the DL-program at hand. Knowledge about such DL-atoms helps to decide whether a DL-program repair exists and it can also be used for optimization purposes. We called such DL-atoms independent and developed a sound and complete calculus for their derivation (Eiter, Fink, and Stepanova 2012). More specifically, independent DL-atoms fall into two categories: tautologic and contradictory. We have shown that checking whether a given DL-atom is independent can be done efficiently.

Moreover, on the theoretical level we have formalized the problem of repairing DL-programs and introduced the notions of repair and repair answer set (Eiter, Fink, and Stepanova 2013). We assumed that the rule part of the DL-program and the ontology TBox are well-developed (as it indeed often happens) and the reasons for inconsistencies lie in the ontology ABox. The novel notions of repair and repair answer set are therefore based on changes of the ontology data part that enable answer sets. For instance, deletion of hasParent(john, pat) from A in Example 1 leads to a repair \( A' = \{ \text{Male}(pat), \text{Male}(john) \} \) under which \( I' = \{ \text{ischildof}(john, alex), \text{boy}(john) \} \) is an flp-repair answer set.

We have shown that repair answer sets do not have higher complexity than ordinary ones (more specifically, weak and FLP answer sets) in case if queries in DL-atoms can be evaluated in polynomial time. To ensure this property, we concentrated on the Description Logic DL-Lite \( A \) (Calvanese et al. 2007), which is a prominent DL particularly useful for ontology based data access (OBDA).

As clearly not all repairs are equally attractive for a given scenario, in order to distinguish between the repairs we introduced the preference relation realized by a selection function \( \sigma \). The latter selects preferred repairs from a set of all candidates. We studied selection functions that do not introduce additional complexity for computing preferred repair answer sets, e.g. bounded \( \delta^+ \)-change, deletion, addition under bounded opposite polarity and others. For instance, while \( A' \) from above is a deletion repair for \( I \), the ABox \( A'' = \{ \text{Male}(pat), \text{Male}(john), \text{hasParent}(john, pat), \text{female}(alex) \} \) satisfies the criteria for being a \( k \)-bouded addition repair, if \( k \geq 1 \). The repair \( A'' \) makes \( I'' = I' \cup \{ \text{hasfather}(john, pat) \} \) a repair answer set for \( I \).

We showed how an algorithm for evaluating DL-programs (Eiter et al. 2005) can be extended to compute repairs resp. repair answer sets, with possibly integrated selection criteria. The evaluation of the DL-program \( II \) is based on the program rewriting \( II \), where DL-atoms are substituted by normal atoms and additional guessing rules on their values are added to the program. The answer set of the rewritten program is also an answer set of the original DL-program if the real values of all DL-atoms coincide with the guessed ones and the minimality check succeeds. While adapting this approach for the repair computation a novel interesting generalized ontology repair problem (ORP) was introduced. The latter is based on an answer set candidate and DL-atoms of the program. The solution to an ORP is an ABox which ensures simultaneous entailment and non-entailment of sets of queries under possible updates.

The naive implementation of the repair answer set computation goes through all answer sets \( I \) of the replacement program \( II \) and checks all possible ABoxes to see whether under any of them \( I \) becomes an answer set of \( II \). While natural this approach does not to scale well for practical application since the number of ABoxes to be checked might be large in general.

Therefore, we proposed an alternative improved approach for repair computation which is based on the notion of support sets. Intuitively, a support set for a ground DL-atom \( a = \text{DL}\{\lambda; Q(t)\} \) is a part of its input which together with the ontology TBox is sufficient for \( Q(t) \) to be derived. Our method is to precompute small support sets for all DL-atoms on a nonground level by exploiting TBox classification. Then for each candidate interpretation the ground instantiations of support sets are effectively obtained. These help to prune the search space of the model candidates and also to construct the ABox repair. The described approach is particularly attractive for DL-Lite \( A \) ontologies, for which support sets are small and easily computable.

We implemented the above algorithm as part of the DLVHEX system and evaluated it on a number of benchmark scenarios. The results of the experiments proved effectiveness of our approach.

5 Open Issues and Expected Achievements

For future work in remains to improve the implementation of the inconsistency framework to enable computing different \( \sigma \)-preferred repairs for DL-programs. We plan to consider maximal deletion repairs, repairs with at most \( k \) new assertions, discussed in (Eiter, Fink, and Stepanova 2013). Moreover, we will realize other practically attractive criteria for repair selection. For example, one might be interested in deleting only assertions with certain ontology predicates or to restrict repairs to selected individuals.

So far we have focused on DL-programs with ontologies in DL-Lite \( A \). Another issue for further research is to look at other DLs, such as \( E \).

Even though the evaluation of the developed inconsistency framework has been already started, a further in-depth analysis is still required. The latter involves creation of benchmarks from real world applications, which is nontrivial since no compared benchmarks exist.

Acknowledgements I would like to thank my supervisor Prof. Dr. Thomas Eiter and Dr. Michael Fink for their ongoing support.
References


Eiter, T.; Fink, M.; and Stepanova, D. 2013. Data repair of inconsistent dl-programs. In JELIA. IJCAI/AAAI.


